N O V A S - C

Naval Observatory Vector Astrometry Subroutines
C Language Version

W. T. Harris
J. A. Bangert
U. S. Naval Observatory

Based on algorithms and Fortran version by:
G. H. Kaplan
U. S. Naval Observatory

First Fortran version: 20 Oct 1988
Revised: 15 Mar 1990
C Version: 7 June 1996

NOVAS-C
Naval Observatory Vector Astrometry Subroutines
C Language Version

William T. Harris
John A. Bangert
U. S. Naval Observatory

Based on the algorithms and Fortran version by:
George H. Kaplan
U. S. Naval Observatory

1. Introduction

The first version of NOVAS, the **N**aval **O**bservatory **V**ector **A**strometry **S**ubroutines, was released in 1988. It was followed by a revised version, still the current version, in 1990. NOVAS is provided in the form of Fortran source code. The Fortran package has proven to be very popular, but over the years, there have been numerous requests for a C-language version. In the early 1990s, members of the U.S. Naval Observatory/Naval Research Laboratory Optical Interferometer group converted parts of NOVAS to C for use in their project. Their work was returned to the Naval Observatory's Astronomical Applications Department for further development. The result is a package of C-language source code called NOVAS-C.

Like its Fortran counterpart, NOVAS-C is an integrated package of modules for the computation of a wide variety of common astrometric quantities and transformations. The modules are all coded in ANSI-standard C. The package can provide, in one function call, the instantaneous coordinates (apparent, topocentric, or astrometric place) of any star or planet. At a lower level, NOVAS-C also provides general astrometric utility transformations, such as those for precession, nutation, aberration, parallax, and the gravitational deflection of light. The computations are precise to better than one milliarcsecond. The NOVAS-C package is an easy-to-use facility which can be incorporated into data reduction programs, telescope control systems, and simulations. The NOVAS-C algorithms are, in fact, virtually identical to those now used in the production of the *Astronomical Almanac*.

The algorithms used by the NOVAS-C functions are based on a vector and matrix formulation which is rigorous, consistent with recent IAU resolutions, and does not use spherical trigonometry or form "day numbers" at any point. Objects within and outside the solar system are treated similarly and the position vectors formed and operated on by these routines place each relevant object at its actual distance (in AU) from the solar system barycenter. Objects at unknown distance (parallax zero or undetermined) are placed on the "celestial sphere" herein defined to be at a radius of 10 megaparsecs (2.06×10^{12} AU). A description of the algorithms used, along with definitions of terms and related information, can be found in Kaplan, et al. (1989) Astron. J. **97**, 1197.

Since the algorithms used in the NOVAS-C functions are consistent with the IAU J2000.0 system, any reference data which the functions require as input, such as a star's catalog mean place and proper motion, must be expressed in this system. A large body of reference data now exists within the IAU J2000.0 system, including the FK5 star catalog, the JPL planetary ephemerides, the ACRS (a replacement for the SAO star catalog), and an ever-expanding set of catalogs of radio sources and other objects.

Three levels of functions are involved: basic, utility, and supervisory. *Basic*-level functions supply the values of fundamental variables, such as the nutation angles and the heliocentric positions of solar system bodies, for specific epochs. *Utility*-level functions perform transformations corresponding to precession, nutation, aberration, etc. *Supervisory*-level functions call the basic and utility functions in the proper order to compute apparent, topocentric, or astrometric places of stars or solar system bodies for specific dates and times. If desired, the user can interact exclusively with the supervisory-level routines and not become concerned with the details of the geometry or physical models involved in the computation.

The NOVAS-C source code contains sufficient internal documentation to make the usage clear. Expanded explanations of a few of the most frequently-called functions are given elsewhere in this document. In the Fortran version of NOVAS, some of the basic- and utility-level subroutines are provided in several versions to accommodate users with a need for alternative algorithms. The C version differs from the Fortran version in this regard: only the “standard” version of each algorithm is provided. However, two versions of the software that provides basic lunar and planetary ephemeris data are included.

The next section of this document (Section 2) provides an overview of the files that constitute NOVAS-C. This section also provides simple instructions for installing and checking the software. Section 3 provides a list and brief description of each NOVAS-C function. Section 4 gives detailed descriptions of a few of the most frequently-called functions. Throughout this document, **bold** text will be used to refer to file names and *italic* text will be used to refer to function or subroutine names. Variable names or code snippets will be presented in a `typewriter-like` font.

Also, be aware that the U. S. Naval Observatory’s Astronomical Application Department maintains a NOVAS page on the World Wide Web. It can be reached via the USNO home page (http://www.usno.navy.mil).

2. File Overview and Installation

The following files make up the NOVAS-C system:

File name	Description
novas.c	contains all supervisory and utility functions and most basic functions
novas.h	header file for **novas.c** (includes structure definitions and function prototypes)
novascon.c	contains most mathematical and physical constants used by the NOVAS-C system
novascon.h	header file for **novascon.c**
solsys2d.c	version of function *solarsystem* that serves as an interface between NOVAS-C and the JPL lunar and planetary ephemerides (see detailed discussion in Section 4)
solsys3.c	version of function *solarsystem* that provides the position and velocity of the Earth or Sun without reference to an external data file (see detailed discussion in Section 4)

In addition, the following files are provided to assist in validating the installation of NOVAS-C on your local system:

checkout.c	main function that calls functions in **novas.c** and **solsys3.c** for the purpose of validating a local installation
checkout.no	output from the "checkout" application computed at USNO; compare this file with results obtained from your local installation

To install NOVAS-C on your local system, do the following:

• Copy all NOVAS-C files to a directory on your local system.

• Compile and link files **checkout.c**, **novas.c**, **novascon.c**, and **solsys3.c**. Name the resulting application "checkout".

• Run the checkout application. Compare the results that you get (in the file **checkout.out**) with the data in file **checkout.no**. If the results agree, the installation has probably been successful.

Important Note

The checkout application exercises one supervisory function and most, but not all, of the low-level functions in **novas.c**. Also, the checkout application does not use **solsys2d.c**; hence, planetary positions (other than those of the Earth) are not tested. Thus, use of the checkout application is not a complete test of NOVAS-C. A more complete check of your NOVAS-C implementation can be made by comparing the results from the NOVAS-C supervisory functions with results from the analogous NOVAS Fortran supervisory functions.

3. Function Overview

The following functions are contained in file **novas.c**:

Entry name	Level	Purpose
app_star	supervisory	Computes the geocentric apparent place of a star, given its J2000.0 catalog mean place.
topo_star	supervisory	Computes the topocentric apparent place of a star, given its J2000.0 catalog mean place and geographic location of observer.
app_planet	supervisory	Computes the geocentric apparent place of a planet or other solar system body.
topo_planet	supervisory	Computes the topocentric apparent place of a planet or other solar system body, given geographic location of observer.
virtual_star	supervisory	Computes the “virtual place” of a star, given its J2000.0 catalog mean place.
local_star	supervisory	Computes the “local place” of a star, given its J2000.0 catalog mean place and geographic location of observer.
virtual_planet	supervisory	Computes the “virtual place” of a planet or other solar system body.
local_planet	supervisory	Computes the “local place” of a planet or other solar system body, given geographic location of observer.
astro_star	supervisory	Computes the astrometric place of a star, given its J2000.0 catalog mean place.
astro_planet	supervisory	Computes the astrometric place of a planet or other solar system body.
mean_star	supervisory	Computes the J2000.0 mean place of a star, given its apparent place.
sidereal_time	supervisory	Computes Greenwich sidereal time, either mean or apparent.
pnsw	supervisory	Transforms arbitrary vector in rotating Earth-fixed (geographic) system to space-fixed (J2000.0) system.
get_earth	utility	Provides barycentric and heliocentric position and velocity of the Earth at a TDT date.
spin	utility	Rotates vector by angle equal to sidereal time.
wobble	utility	Adjusts Earth-fixed vector for polar motion.
proper_motion	utility	Updates the position vector of a star to allow for its space motion.
geocentric	utility	Changes origin of coordinates from barycenter of solar system to center of mass of Earth.
aberration	utility	Adjusts position vector for aberration of light due to motion of Earth.
precession	utility	Applies precession to position vector.
nutate	utility	Applies nutation to position vector.
sun_field	utility	Adjusts position vector for deflection of light by Sun’s gravitational field.
terra	utility	Converts geographic coordinates to geocentric position vector.
vector2radec	utility	Converts position vector to RA and declination.
angle2vector	utility	Converts RA, declination, and distance to a position vector.
starvectors	utility	Converts RA, declination, proper motion, etc., to position and velocity vectors.
calc_nutation	basic	Evaluates nutation series.
earth_tilt	basic	Provides information on orientation of Earth’s axis: obliquity, nutation parameters, etc.
convert_tdb2tdt	basic	Converts Terrestrial Dynamical Time (TDT) to Barycentric Dynamical Time (TDB).

In order to compute instantaneous positions using the supervisory functions, NOVAS-C must have access to a solar system ephemeris. The solar system ephemeris provides NOVAS-C with the heliocentric and barycentric positions and velocities of desired solar system objects referred to the mean equator and equinox of J2000.0. The solar system ephemeris is required even when only precise star positions are needed – in that case, the "desired solar system object" is the Earth.

NOVAS-C accesses the ephemeris through a function called *solarsystem*. While this function has a defined argument list, its inner workings can take any form depending upon the ephemeris that has been selected for use. Users may write their own versions of *solarsystem* or use either of the two versions provided with NOVAS-C:

• **solsys2d.c** serves as the interface between NOVAS-C and the JPL lunar and planetary ephemerides, such as DE200 or DE403. This function contains a single call to JPL's Fortran subroutine *pleph*, which in turn calls other Fortran subroutines in the JPL ephemeris software package. The user must obtain the Fortran ephemeris package from JPL, set up the binary, random-access ephemeris file, and link the applicable JPL Fortran code with NOVAS-C. For more details, see the discussion of this version of *solarsystem* in Section 4.

• **solsys3.c** provides the position and velocity of the Earth or Sun without reference to an external data file. This version of *solarsystem* is ideally suited for computing coordinates of stars, with errors not exceeding several milliarcseconds.

See the next section for additional information on function *solarsystem* and other frequently called functions.

4. Important Functions in NOVAS-C

APP_STAR

```
short int app_star (double tjd, short int earth, fk5_entry *star,

                    double *ra, double *dec)
```

```
  PURPOSE:
      Computes the apparent place of a star at date 'tjd', given its
      mean place, proper motion, parallax, and radial velocity for
      J2000.0.

  INPUT
   ARGUMENTS:
      tjd (double)
         TDT Julian date for apparent place.
      earth (short int)
         Body identification number for the Earth.
      *star (struct fk5_entry)
         Pointer to catalog entry structure (defined in novas.h).

   OUTPUT
   ARGUMENTS:
      *ra (double)
         Apparent right ascension in hours, referred to true equator
         and equinox of date 'tjd'.
      *dec (double)
         Apparent declination in degrees, referred to true equator
         and equinox of date 'tjd'.

   RETURNED
   VALUE:
      (short int)
          0...Everything OK.
         >0...Error code from function 'solarsystem'.
```

Discussion:

This function computes the apparent place of a star. The word "star" as used here refers to any object outside the solar system. If the values of `promora`, `promodec`, `parlax`, or `radvel` within structure `star` are unknown (or zero within the errors of measurement), the calling program should set them to zero. For extragalactic objects, these input values should be set to zero. The user's choice of the version of function *solarsystem* determines the value of the argument `earth` that the calling program must supply to *app_star*.

The input mean place at standard epoch J2000.0 is assumed to be the true mean place, similar to the mean places in the FK5, and not contain the so-called "E-terms".

Efficiency is maximized when successive calls to *app_star* have the same value for `tjd`, since some quantities which are functions only of time are thereby saved and reused.

TOPO_STAR

```
short int topo_star (double tjd, short int earth, double deltat,
                     fk5_entry *star, site_info *location,

                     double *ra, double *dec)


   PURPOSE:
      Computes the topocentric place of a star at date 'tjd', given its
      mean place, proper motion, parallax, and radial velocity for
      J2000.0 and the location of the observer.

  INPUT
   ARGUMENTS:
      tjd (double)
         TDT Julian date for topocentric place.
      earth (short int)
         Body identification number for the Earth.
      deltat (double)
         Difference TDT-UT1 at 'tjd', in seconds.
      *star (struct fk5_entry)
         Pointer to catalog entry structure (defined in novas.h).
      *location (struct site_info)
         Pointer to structure containing observer's location (defined
         in novas.h).

   OUTPUT
   ARGUMENTS:
      *ra (double)
         Topocentric right ascension in hours, referred to true equator
         and equinox of date 'tjd'.
      *dec (double)
         Topocentric declination in degrees, referred to true equator
         and equinox of date 'tjd'.

   RETURNED
   VALUE:
      (short int)
          0...Everything OK.
         >0...Error code from function 'solarsystem'.
```

Discussion:

This function computes the topocentric place of a star (neglecting atmospheric refraction) for the location specified by the argument `location`, for the time specified by the argument `tjd`. Note that `tjd` is the TDT time at which the topocentric place is to be computed. The word “star” as used here refers to any object outside the solar system. If the values of `promora`, `promodec`, `parlax`, or `radvel` within structure `star` are unknown (or zero within the errors of measurement), the calling program should set them to zero. For extragalactic objects, these input values should be set to zero. The difference TDT–UT1 (often called ΔT) is passed to the function via argument `deltat`. Values of ΔT are published in the annual *Astronomical Almanac* or can be obtained from the National Earth Orientation Service (NEOS) home page on the World Wide Web (http://maia.usno.navy.mil/). The user’s choice of the version of function *solarsystem* determines the value of the argument `earth` that the calling program must supply to *app_star*.

The input mean place at standard epoch J2000.0 is assumed to be the true mean place, similar to the mean places in the FK5, and not contain the so-called “E-terms”.

APP_PLANET

```
short int app_planet (double tjd, short int planet, short int earth,

                      double *ra, double *dec, double *dis)
```

```
   PURPOSE:
      Compute the apparent place of a planet or other solar system body.

  INPUT
   ARGUMENTS:
      tjd (double)
         TDT Julian date for apparent place.
      planet (short int)
         Body identification number for desired planet.
      earth (short int)
         Body identification number for the Earth.

   OUTPUT
   ARGUMENTS:
      *ra (double)
         Apparent right ascension in hours, referred to true equator
         and equinox of date 'tjd'.
      *dec (double)
         Apparent declination in degrees, referred to true equator
         and equinox of date 'tjd'.
      *dis (double)
         True distance from Earth to planet at 'tjd' in AU.

   RETURNED
   VALUE:
      (short int)
          0...Everything OK.
         >0...Error code from function 'solarsystem'.
```

Discussion:

This function computes the apparent place of a planet or other solar system body by calling function *solarsystem* to obtain its rectangular coordinates, along with those of the Earth. Other utility- and basic-level functions are also called. The user's choice of the version of *solarsystem* to be used determines the values of the arguments `planet` and `earth`, which identify the planet and the Earth, respectively. The source of the rectangular coordinates is, of course, also determined by the version of *solarsystem* in use.

Efficiency is maximized when successive calls to *app_planet* have the same value for `tjd`, since some quantities which are functions only of time are thereby saved and reused.

TOPO_PLANET

```
short int topo_planet (double tjd, short int planet, short int earth,
                       double deltat, site_info *location,

                       double *ra, double *dec, double *dis)
```

```
   PURPOSE:
      Computes the topocentric place of a planet, given the location of
      the observer.

  INPUT
   ARGUMENTS:
      tjd (double)
         TDT Julian date for topocentric place.
      planet (short int)
         Body identification number for desired planet.
      earth (short int)
         Body identification number for the Earth.
      deltat (double)
         Difference TDT-UT1 at 'tjd', in seconds.
      *location (struct site_info)
         Pointer to structure containing observer's location (defined
         in novas.h).

   OUTPUT
   ARGUMENTS:
      *ra (double)
         Topocentric right ascension in hours, referred to true
         equator and equinox of date 'tjd'.
      *dec (double)
         Topocentric declination in degrees, referred to true equator
         and equinox of date 'tjd'.
      *dis (double)
         True distance from observer to planet at 'tjd' in AU.

   RETURNED
   VALUE:
      (short int)
          0...Everything OK.
         >0...Error code from function 'solarsystem'.
```

Discussion:

This function computes the topocentric place of a planet or other solar system body (neglecting atmospheric refraction) for the location specified by the argument `location`, for the time specified by the argument `tjd`. Note that `tjd` is the TDT time at which the topocentric place is to be computed. The difference TDT–UT1 (often called ΔT) is passed to the function via argument `deltat`. Values of ΔT are published in the annual *Astronomical Almanac* or can be obtained from the National Earth Orientation Service (NEOS) home page on the World Wide Web (http://maia.usno.navy.mil/). The user's choice of the version of *solarsystem* determines the values of the arguments `planet` and `earth`, which identify the planet and the Earth, respectively. The source of the rectangular coordinates is, of course, also determined by the version of *solarsystem* in use.

VIRTUAL_STAR

```
short int virtual_star (double tjd, short int earth, fk5_entry *star,
                        double *ra, double *dec)
```

```
   PURPOSE:
      Computes the virtual place of a star at date 'tjd', given its
      mean place, proper motion, parallax, and radial velocity for
      J2000.0.

  INPUT
   ARGUMENTS:
      tjd (double)
         TDT Julian date for virtual place.
      earth (short int)
         Body identification number for the earth.
      *star (struct fk5_entry)
         Pointer to catalog entry structure (defined in novas.h).

   OUTPUT
   ARGUMENTS:
      *ra (double)
         Virtual right ascension in hours, referred to mean equator
         and equinox of J2000.
      *dec (double)
         Virtual declination in degrees, referred to mean equator
         and equinox of J2000.

   RETURNED
   VALUE:
      (short int)
          0...Everything OK.
         >0...Error code from function 'solarsystem'.
```

Discussion:

See the discussion for function *app_star*. Function *virtual_star* is identical to *app_star* in input arguments and use. Here, however, the output arguments provide the virtual place of the star. The virtual place is essentially the apparent place expressed in the coordinate system of standard epoch J2000.0.

LOCAL_STAR

```
short int local_star (double tjd, short int earth, double deltat,
                      fk5_entry *star, site_info *location,

                      double *ra, double *dec)


   PURPOSE:
      Computes the local place of a star, given its mean place, proper
      motion, parallax, and radial velocity for J2000.0, and the
      location of the observer.

  INPUT
   ARGUMENTS:
      tjd (double)
         TDT Julian date for local place.
      earth (short int)
         Body identification number for the Earth.
      deltat (double)
         Difference TDT-UT1 at 'tjd', in seconds.
      *star (struct fk5_entry)
         Pointer to catalog entry structure (defined in novas.h).
      *location (struct site_info)
         Pointer to structure containing observer's location (defined
         in novas.h).

   OUTPUT
   ARGUMENTS:
      *ra (double)
         Local right ascension in hours, referred to mean equator and
         equinox of J2000.
      *dec (double)
         Local declination in degrees, referred to mean equator and
         equinox of J2000.

   RETURNED
   VALUE:
      (short int)
          0...Everything OK.
         >0...Error code from function 'solarsystem'.
```

Discussion:

See the discussion for function *topo_star*. Function *local_star* is identical to *topo_star* in input arguments and use. The local place is essentially the topocentric place expressed in the coordinate system of standard epoch J2000.0.

VIRTUAL_PLANET

```
short int virtual_planet (double tjd, short int planet, short int earth,
                          double *ra, double *dec, double *dis)
```

```
   PURPOSE:
      Computes the virtual place of a planet or other solar system body.

  INPUT
   ARGUMENTS:
      tjd (double)
         TDT Julian date for virtual place.
      earth (short int)
         Body identification number for the Earth.
      planet (short int)
         Body identification number for desired planet.

   OUTPUT
   ARGUMENTS:
      *ra (double)
         Virtual right ascension in hours, referred to mean equator
         and equinox of J2000.
      *dec (double)
         Virtual declination in degrees, referred to mean equator
         and equinox of J2000.
      *dis (double)
         True distance from Earth to planet in AU.

   RETURNED
   VALUE:
      (short int)
          0...Everything OK.
         >0...Error code from function 'solarsystem'.
```

Discussion:

See the discussion for function *app_planet*. Function *virtual_planet* is identical to *app_planet* in input arguments and use. Here, however, the output arguments provide the virtual place of the planet. The virtual place is essentially the apparent place expressed in the coordinate system of standard epoch J2000.0.

LOCAL_PLANET

```
short int local_planet (double tjd, short int planet, short int earth,
                        double deltat, site_info *location,

                        double *ra, double *dec, double *dis)


   PURPOSE:
      Computes the local place of a planet or other solar system body,
      given the location of the observer.

  INPUT
   ARGUMENTS:
      tjd (double)
         TDT Julian date for local place.
      earth (short int)
         Body identification number for the Earth.
      planet (short int)
         Body identification number for desired planet.
      deltat (double)
         Difference TDT-UT1 at 'tjd', in seconds.
      *location (struct site_info)
         Pointer to structure containing observer's location (defined
         in novas.h).

   OUTPUT
   ARGUMENTS:
      *ra (double)
         Local right ascension in hours, referred to mean equator and
         equinox of J2000.
      *dec (double)
         Local declination in degrees, referred to mean equator and
         equinox of J2000.
      *dis (double)
         True distance from Earth to planet in AU.

   RETURNED
   VALUE:
      (short int)
          0...Everything OK.
         >0...Error code from function 'solarsystem'.
```

Discussion:

See the discussion for function *topo_planet*. Subroutine *local_planet* is identical to *topo_planet* in input arguments and use. The local place is essentially the topocentric place expressed in the coordinate system of standard epoch J2000.0.

ASTRO_STAR

```
short int astro_star (double tjd, short int earth, fk5_entry *star,

                      double *ra, double *dec)
```

```
   PURPOSE:
      Computes the astrometric place of a star, given its mean place,
      proper motion, parallax, and radial velocity for J2000.0.

  INPUT
   ARGUMENTS:
      tjd (double)
         TDT Julian date for astrometric place.
      earth (short int)
         Body identification number for the Earth.
      *star (struct fk5_entry)
         Pointer to catalog entry structure (defined in novas.h).

   OUTPUT
   ARGUMENTS:
      *ra (double)
         Astrometric right ascension in hours, referred to mean equator
         and equinox of J2000.
      *dec (double)
         Astrometric declination in degrees, referred to mean equator
         and equinox of J2000.

   RETURNED
   VALUE:
      (short int)
          0...Everything OK.
         >0...Error code from function 'solarsystem'.
```

Discussion:

See the discussion for function *app_star*. Function *astro_star* is identical to *app_star* in input arguments and use. Here, however, the output arguments provide the astrometric place of the star.

ASTRO_PLANET

```
short int astro_planet (double tjd, short int planet, short int earth,
                        double *ra, double *dec, double *dis)
```

```
   PURPOSE:
      Computes the astrometric place of a planet or other solar system
      body.

  INPUT
   ARGUMENTS:
      tjd (double)
         TDT Julian date for calculation.
      planet (short int)
         Body identification number for desired planet.
      earth (short int)
         Body identification number for the Earth.

   OUTPUT
   ARGUMENTS:
      *ra (double)
         Astrometric right ascension in hours, referred to mean equator
         and equinox of J2000.
      *dec (double)
         Astrometric declination in degrees, referred to mean equator
         and equinox of J2000.
      *dis (double)
         True distance from Earth to planet in AU.

   RETURNED
   VALUE:
      (short int)
          0...Everything OK.
         >0...Error code from function 'solarsystem'.
```

Discussion:

See the discussion for function *app_planet*. Function *astro_planet* is identical to *app_planet* in input arguments and use. Here, however, the output arguments provide the astrometric place of the planet.

SIDEREAL_TIME

```
void sidereal_time (double julianhi, double julianlo, double ee,

                    double *gst)
```

```
   PURPOSE:
      Computes the Greenwich apparent sidereal time, at Julian date
      'julianhi' + 'julianlo'.

   INPUT
   ARGUMENTS:
      julianhi (double)
         Julian date, integral part.
      julianlo (double)
         Julian date, fractional part.
      ee (double)
         Equation of the equinoxes (seconds of time)

   OUTPUT
   ARGUMENTS:
      *gst (double)
         Greenwich apparent sidereal time, in hours.

   RETURNED
   VALUE:
      None.
```

Discussion:

This function computes Greenwich sidereal time. To obtain the Greenwich mean sidereal time, set input argument `ee` = 0.0. To obtain Greenwich apparent sidereal time, supply the correct value for the equation of the equinoxes (`ee`) which can be computed by calling function *earthtilt*.

The input Julian date may be split into two parts to ensure maximum precision in the computation. For maximum precision, `julianhi` should be set to be equal to the integral part of the Julian date, and `julianlo` should be set to be equal to the fractional part. For most applications the position of the split is not critical as long as the sum `julianhi + julianlo` is correct: for example, when used with computers providing 16 decimal digits of precision in `double` variables, this function will yield values of `gst` precise to better than 1 millisecond even if `julianhi` contains the entire Julian date and `julianlo` is set to 0.0.

For most uses, the input Julian date should be in the UT1 time scale. If the input Julian date is in the TDB time scale, the output must be considered to be ‘dynamical’ sidereal time.

PRECESSION

```
short int precession (double tjd1, double *pos, double tjd2,

                      double *pos2)
```

```
   PURPOSE:
      Precesses equatorial rectangular coordinates from one epoch to
      another.  The coordinates are referred to the mean equator and
      equinox of the two respective epochs.

  INPUT
   ARGUMENTS:
      tjd1 (double)
         TDB Julian date of first epoch.
      pos[3] (double)
         Position vector, geocentric equatorial rectangular coordinates,
         referred to mean equator and equinox of first epoch.
      tjd2 (double)
         TDB Julian date of second epoch.

   OUTPUT
   ARGUMENTS:
      pos2[3] (double)
         Position vector, geocentric equatorial rectangular coordinates,
         referred to mean equator and equinox of second epoch.

   RETURNED
   VALUE:
      (short int)
         0...Everything OK.
```

Discussion:

This function precesses a position vector `pos1` from the equatorial rectangular system of epoch `tjd1` to the equatorial rectangular system of epoch `tjd2`; the resulting vector is `pos2`. The two epochs are completely arbitrary and the transformation is reversible. In typical usage, one of the two epochs will be standard epoch J2000.0, that is, either `tjd1` or `tjd2` will be 2451545.0.

EARTHTILT

```
void earthtilt (double tjd,

                double *mobl, double *tobl, double *eq, double *dpsi,
                double *deps)


   PURPOSE:
      Computes quantities related to the orientation of the Earth's
      rotation axis at Julian date 'tjd'.

  INPUT
   ARGUMENTS:
      tjd (double)
         TDB Julian date of the desired time

   OUTPUT
   ARGUMENTS:
      *mobl (double)
         Mean obliquity of the ecliptic in degrees at 'tjd'.
      *tobl (double)
         True obliquity of the ecliptic in degrees at 'tjd'.
      *eq (double)
         Equation of the equinoxes in seconds of time at 'tjd'.
      *dpsi (double)
         Nutation in longitude in seconds of arc at 'tjd'.
      *deps (double)
         Nutation in obliquity in seconds of arc at 'tjd'.

   RETURNED
   VALUE:
      None.
```

Discussion:

This function computes various quantities related to the orientation of the Earth's rotation axis in inertial space at a specific time. The computation involves a call to function *calcnutation* to evaluate the nutation series.

SOLARSYSTEM

```
short int solarsystem (double tjd, short int body, short int origin,
                      double *pos, double *vel)
```

```
PURPOSE:
   Provides the position and velocity vectors of a planet or other
   solar system body at a specific time. The origin of coordinates
   may be either the barycenter of the solar system or the center
   of mass of the Sun.

INPUT
ARGUMENTS:
   tjd (double)
      TDB Julian date.
   body (short int)
      Body identification number for the solar system object of
      interest;  Mercury = 1,...,Pluto = 9, Sun = 10, Moon = 11.
   origin (short int)
      Origin code; solar system barycenter   = 0,
                   center of mass of the Sun = 1.

OUTPUT
ARGUMENTS:
   pos[3] (double)
      Position vector of 'body' at tjd; equatorial rectangular
      coordinates in AU referred to the mean equator and equinox
      of J2000.0.
   vel[3] (double)
      Velocity vector of 'body' at tjd; equatorial rectangular
      system referred to the mean equator and equinox of J2000.0,
      in AU/Day.

RETURNED
VALUE:
   (short int)
      0...Everything OK.
      Other values depend upon version in use
```

Discussion:

This subroutine supplies values for the components of the position vector `pos` and velocity vector `vel` for body `body` at time `tjd`. The vectors computed by *solarsystem* are in the equatorial rectangular coordinate system which is oriented to the mean equator and equinox of standard epoch J2000.0. The vectors are barycentric if `origin`=0 and heliocentric if `origin`=1.

There are two different versions of *solarsystem* supplied in NOVAS-C, each with its own internal logic. One uses internally-stored data or series expansions, the other refers to external data files. Additional documentation is provided on the following pages for the proper use of each version. The user is free to supply alternative versions, providing that the arguments conform to the above specifications.

The values of the body identification number, `body`, will in general differ from one *solarsystem* version to another; consult the documentation for the specific version in use. Usually, `body`=1 refers to Mercury, `body`=2 refers to Venus, `body`=3 refers to the Earth, etc., but the identification numbers for bodies such as the Sun or Moon vary. Furthermore, some versions of *solarsystem* support only a subset of the major solar system bodies. *The minimum requirement is support for the Earth.* It is also sometimes necessary to distinguish between the Earth and the Earth/Moon barycenter; for computing quantities related to observables (e.g., apparent, topocentric, or astrometric places) it is the position and velocity of the Earth that is required.

```
RETURNED
VALUE:
   (short int)
      0...Everything OK.
      1...Invalid value of body or origin.
      2...Error detected by JPL software.
```

Discussion

This version serves as the interface between the Jet Propulsion Laboratory's lunar and planetary ephemeris software and NOVAS-C. The function contains a single call to JPL's Fortran subroutine *pleph*, which in turn calls other Fortran subroutines in the JPL ephemeris software package. The user is responsible for obtaining the Fortran ephemeris package from JPL, setting up the binary, random-access ephemeris file, and linking the JPL Fortran with NOVAS-C. See the Implementation Notes below.

The body identification numbers to be used with this version are: Sun, `body`=10; Mercury, `body`=1; Venus, `body`=2; Earth, `body`=3; Mars, `body`=4; Jupiter, `body`=5; Saturn, `body`=6; Uranus, `body`=7; Neptune, `body`=8; Pluto, `body`=9; Moon, `body`=11.

Implementation Notes

In order to use NOVAS-C with *solarsystem* version 2D, you must first obtain the export planetary ephemeris package from JPL. Be sure to choose an ephemeris whose coordinates are oriented to the mean equator and equinox of standard epoch J2000.0, such as DE200 or DE403. The export package is available over the Internet from the anonymous ftp server navigator.jpl.nasa.gov/ephem/export and consists of several large ASCII data files and software provided in the form of Fortran source code. An installation guide is also included. The installation process consists of converting the (large) file of ASCII ephemeris data to binary, direct-access form using a supplied utility program. Then, the binary file is verified using another utility program and a file of comparison data. If the verification process is successful, the ephemeris file is ready to use. The ephemeris data is obtained from the binary file by calling the access subroutines provided in the export package.

Important Note

Over the years, there have been several versions of the JPL export ephemeris software. The following discussion specifically refers to the software version documented in "The JPL Export Planetary Ephemeris", 29 June 1990 revision 1, by E. M. Standish and X X Newhall of JPL. Use of *solarsystem* version 2D with other versions of the JPL software will require minor modifications to the *solarsystem* source code.

Version 2D of *solarsystem* (in C) obtains ephemeris data from the binary file by calling JPL subroutine *pleph* (in Fortran). The C function has a few features that make it possible for it to exchange data with the Fortran subroutine. First, all of the arguments of the call to *pleph* in the function are addresses, since Fortran uses call by address instead of call by value, for arguments of subroutines. Second, all of the integer arguments in the call are designated as type `long int` in the C function in an attempt to match the Fortran `INTEGER` default. Finally, the "alternate return" asterisk argument (last item in the *pleph* argument list) is designated as a `long int` in the C function; the returned value is used as an error flag. The `DOUBLE PRECISION` arguments in the subroutine are designated as type `double` in the C function.

Probably the biggest hurdle in implementing version 2D of *solarsystem* will involve the proper compiling and linking of the mixed-language files. The procedures will be specific to your computing platform; therefore, you will have to consult your compiler manual for detailed instructions. The following instructions are offered only as a guideline – they provides a specific example of how the mixed-language files were handled successfully on an IBM RISC System 6000 Unix workstation.

1. Create a single file with all of the JPL Fortran ephemeris access subroutines. Name it `jplsubs.f`.

2. Compile the Fortran file without invoking the linkage editor. This creates the object file `jplsubs.o`. The Fortran compiler/linker is `xlf`.

```
xlf -c jplsubs.f
```

3. Compile, again without invoking the linkage editor, the C files `novas.c`, `novascon.c`, `solsys2d.c`. This creates the object files `novas.o`, `novascon.o`, `solsys2d.o`:

```
xlc -c -lm novas.c novascon.c solsys2d.c
```

The C compiler/linker is `xlc`. The `-lm` option specifically searches the math library.

4. Finally, compile the main function and link it with the object files:

object files

search the C math library

main function

```
xlc -L/usr/lib -lm -lxlf90 -o app main.c novas.o novascon.o solsys2d.o jplsubs.o
```

search in this directory
for -l libraries

search the Fortran
libraries (xlf90)

create executable named "app"

In this example, the resulting executable file is named “app.” Note especially the use of the `-l` option to force the C compiler/linker to search the Fortran libraries for unresolved references.

Important Note

It is strongly recommended that results obtained from your specific implementation of NOVAS-C and *solarsystem* version 2D be checked by comparing to corresponding values published in the *Astronomical Almanac*, or by comparing to results obtained from the Fortran version of NOVAS using subroutine `SOLSYS` version 2-DA.

SOLARSYSTEM, Version 3

```
RETURNED
VALUE:
   (short int)
      0...Everything OK.
      1...Input Julian date ('tjd') out of range.
      2...Invalid value of 'body'.
```

Discussion:

This version of *solarsystem* provides the position and velocity of the Earth or Sun without reference to any external data file. The heliocentric position and velocity of the Earth are computed by evaluating trigonometric series. When barycentric positions and velocities are required, a number of somewhat crude approximations are involved; therefore, barycentric positions and velocities computed by this version of SOLSYS are less accurate than heliocentric positions and velocities. The resulting errors should be less than the following values:

Maximum error in heliocentric positions:	6×10^{-6}	AU
Maximum error in heliocentric velocities:	8×10^{-7}	AU/day
Maximum error in barycentric positions:	8×10^{-4}	AU
Maximum error in barycentric velocities:	2×10^{-6}	AU/day

When this version of *solarsystem* is used in the computation of the apparent place of the Sun, it should contribute less than 2 arcseconds error. When this version of *solarsystem* is used in the computation of apparent places of stars, it should contribute less than 2 milliarcseconds error.

The above error assessment applies to the interval 1800–2050.

Note: This version of *solarsystem* calls several other functions in the NOVAS-C package.

The body identification numbers to be used with this version are: Sun, `body`=0, `body`=1, or `body`=10; Earth, `body`=2 or `body`=3.

Acknowledgements

Thomas K. Buchanan, working as part of the U.S. Naval Observatory/Naval Research Laboratory Optical Interferometer team, did the initial conversion of many of the NOVAS Fortran subroutines to C.

David Buscher, James Hilton, Christian Hummel, and Sandra Martinka, users of preliminary versions of the NOVAS-C package, provided valuable comments and suggestions.